UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/818,097	03/26/2001	Stepan Sokolov	SUN1P815/P5613	2836

22434 7590 02/10/2005

BEYER WEAVER & THOMAS LLP
P.O. BOX 70250
OAKLAND, CA 94612-0250

EXAMINER	
STEELMAN, MARY J	
ART UNIT	**PAPER NUMBER**
2122	

DATE MAILED: 02/10/2005

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	09/818,097	SOKOLOV ET AL.	
	Examiner	Art Unit	
	Mary J. Steelman	2122	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *11/15/2004,09/20/2004*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1,4-19 and 21* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1,4-19 and 21* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *05 May 2004* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

1. This Office Action is in response to Amendment dated 20 September 2004 and RCE filed 15 November 2004. Per Applicant's request claims 2, 3 and 20 are canceled. Claims 1, 7, 9, 15, and 19 have been amended. Claims 1, 4-19 and 21 are pending.

Drawings

2. In view of the amendment to Fig. 1, the prior objection is hereby withdrawn.

Specification

3. In view of the amendments to the Specification, the prior objections are hereby withdrawn.

Claim Objections

4. Claim 15, line 11, recites, "...said information directly form said class...", should be -- ...said information directly from said class...-- Change 'form' to 'from'.

Claim Rejections - 35 USC § 112

5. In view of the amendments to the claims, the prior 35 U.S.C. 112, first and second paragraph rejections are hereby withdrawn.

Claim Rejections - 35 USC § 103

6. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

7. Claims 1, 4-12, 15, 17-19 and 21 are rejected under 35 U.S.C. 103(a) as being unpatentable over US patent 6,738,977 B1 to Berry et al., in view of 6,339,841 to Merrick et al.

Per claims 1 and 19:

Berry disclosed:

-method / computer readable media

(Berry: col. 2, line 40, "...provides a method...", col. 4, lines 55, "provides a computer program product...instructions typically recorded onto a storage medium (computer readable media)...")

- loading said class file into a memory portion of the computing system, wherein said loading operates to copy said class file in its entirety into said memory portion prior to further processing said class file;

(Berry: Abstract, lines 2-3, "A class file is loaded by the first virtual machine into shared memory..." Berry disclosed loading (col. 9, lines 27-28) class files into a master virtual machine heap (memory portion) prior to further processing. Further processing is done by selectively loading some of the master class files into secondary virtual machine (client JVM) heaps. Col. 9, lines 58-63, regarding client JVM heaps, "there are important distinctions in the way I which a shared class loader operates in a client JVM...there may be variations in terms of the private class loaders between the master JVM and its various clients (client JVM heaps)." Class files are loaded in entirety into the master heap.)

Berry disclosed a relationship between multiple virtual machines. Classes are loaded into the heap / memory portion of a master virtual machine. Col. 2, lines 58-67, "The invention provides a master (first) virtual machine and at least one client (second) virtual machine running

in parallel on the same computer system...certain class properties...may need to be set individually on each virtual machine." Thus Berry provides motivation for a "special version" of the class to be loaded onto the client / second virtual machine heap / memory portion. Berry disclosed (col. 11, lines 26-33), "...the requirement to mirror all of the method block is platform dependent..." Thus Berry suggested that the second virtual machine may load only a portion of the class.

Merrick provided additional information regarding the selection of information that is to be loaded into the virtual machine. Merrick disclosed:

-selecting information from said class file in said memory portion after said class file has been loaded into said memory portion, wherein said selecting operates to select information that is to be loaded into said virtual machine;

(Merrick disclosed that the class file could be partitioned to allow for dependent methods to be grouped together (selected) for loading. Col. 3, lines 51-54, "Other ways of breaking down a class file are possible, for instance one could group some methods together if they were dependent on one another.")

-loading said selected information from said class file in said memory portion into said virtual machine and not loading information which has not been selected from said class file into said virtual machine.

(Merrick: Col. 4, lines 33-37, "...the modified class loader will be asked to load x.class (step 1) (loading said selected information). Instead of downloading the x.class in its entirety (not

loading information which has not been selected)..." Merrick suggests that only the requested information is loaded. Col. 4, lines 22-27, "...a load method invoker (modified class loader) to retrieve the method component that has not been loaded onto the client..." and col. 4, lines 64-65, "The method byte code is written to the location pointed at by the invoker...")

Therefore, it would have been obvious, to one of ordinary skill in the art, to have modified Berry's invention, to include the selection of information for a selective load of class files into a virtual machine, as further detailed by Merrick, because both references pertain to virtual machine loading of classes, and both disclose a modified load of class code for the purpose of a quicker virtual machine start up. Berry, col. 11, lines 40-41. Merrick, col. 1, lines 31-35, "Time is also saved as only the classes that are needed are loaded..."

Per claims 4 and 21:

-selecting of information operates to select information from said class file that is needed to be used by the virtual machine.

(Merrick: Col. 3, lines 57-59, "The class loader would check for 'compulsory methods' (likely / needed to be used) in the metadata...and load those methods referenced." Also, col. 5, lines 16-17, "...one of the methods within the class was referenced then only the block of data representing this method is loaded...along with the other essential components of the class.")

Therefore, it would have been obvious, to one of ordinary skill in the art, to have modified Berry's invention, to include the selection of information for a selective load of class

files into a virtual machine, as further detailed by Merrick, because both references pertain to virtual machine loading of classes, and both disclose a modified load of class code for the purpose of a quicker virtual machine start up. Berry, col. 11, lines 40-41. Merrick, col. 1, lines 31-35, "Time is also saved as only the classes that are needed are loaded..."

Per claim 5:

- selecting of information operates to select information that includes information associated with at least one method of said class. (Merrick: Col. 3, lines 46-51, "Each individual accessible component is a part of the class that is separable..metadata ...and individual methods...other separable parts...parts of the constant pool...", col. 5, lines 18-19, "...along with other essential components of the class.")

Therefore, it would have been obvious, to one of ordinary skill in the art, to have modified Berry's invention, to include the selection of information for a selective load of class files into a virtual machine, as further detailed by Merrick, because both references pertain to virtual machine loading of classes, and both disclose a modified load of class code for the purpose of a quicker virtual machine start up. Berry, col. 11, lines 40-41. Merrick, col. 1, lines 31-35, "Time is also saved as only the classes that are needed are loaded..." It is inherent that if a class load is modified, that some type of selection criteria is applied.

Per claim 6:

-loading of only said selected information operates to create an internal representation of the class file in the virtual machine. (Merrick: See fig. 2. Client holds and internal representation of the class file.)

Therefore, it would have been obvious, to one of ordinary skill in the art, to have modified Berry's invention, to include the selection of information for a selective load of class files into a virtual machine, as further detailed by Merrick, because both references pertain to virtual machine loading of classes, and both disclose a modified load of class code for the purpose of a quicker virtual machine start up. Berry, col. 11, lines 40-41. Merrick, col. 1, lines 31-35, "Time is also saved as only the classes that are needed are loaded..."

Per claim 7:

-generating in said internal representation of said class file a method reference portion for said selected information, wherein said method reference portion includes one or more references cells which provide information associated with one or more methods which have been selected to be loaded into said virtual machine.

(Merrick: See fig. 2, #26, Method Table. Col. 3, lines 62-64, "During the class loading the client receives the linear sequence of bytes codes (method code field) and reconstructs the class structure." Also, col. 4, line 10-12, "The method table comprises the names of the methods used by the class and links to the methods or method invokers for that method." The method invoker has signature field.)

Therefore, it would have been obvious, to one of ordinary skill in the art, to have modified Berry's invention, to include the selection of information for a selective load of class files into a virtual machine, as further detailed by Merrick, because both references pertain to virtual machine loading of classes, and both disclose a modified load of class code for the purpose of a quicker virtual machine start up. Berry, col. 11, lines 40-41. Merrick, col. 1, lines 31-35, "Time is also saved as only the classes that are needed are loaded..." Although Merrick may not explicitly show 'a method reference portion that includes one or more references cells which provide information associated with one or more methods which have been selected to be loaded into the virtual machine", it is inherent that that type of information is contained in a method table. (See "How the Java virtual machine handles method invocation and return" by JavaWorld (June 1997))

Per claim 8:

-said method reference portion includes a method name field, a method signature field, and a method code field.

(Merrick: col. 4, line 10-12, "The method table comprises the names of the methods used by the class and links to the methods or method invokers for that method." The method invoker has a signature field.)

Therefore, it would have been obvious, to one of ordinary skill in the art, to have modified Berry's invention, to include the selection of information for a selective load of class files into a virtual machine, as further detailed by Merrick, because both references pertain to virtual machine loading of classes, and both disclose a modified load of class code for the

purpose of a quicker virtual machine start up. Berry, col. 11, lines 40-41. Merrick, col. 1, lines 31-35, "Time is also saved as only the classes that are needed are loaded..."

Per claims 9 and 17:

-said loading of only said selected information operates to populate said method name field, method signature field, and method code field with information or references to information.

(Merrick: See fig. 2, #26. Col. 3, lines 62-64, "During the class loading the client receives the linear sequence of bytes codes (method code field) and reconstructs the class structure." Also, col. 4, line 10-12, "The method table comprises the names of the methods used by the class and links to the methods or method invokers for that method." The method invoker has signature field.)

Therefore, it would have been obvious, to one of ordinary skill in the art, to have modified Berry's invention, to include the selection of information for a selective load of class files into a virtual machine, including populating a method table fields with name, signature, and code, as further detailed by Merrick, because both references pertain to virtual machine loading of classes, and both disclose a modified load of class code for the purpose of a quicker virtual machine start up. Berry, col. 11, lines 40-41. Merrick, col. 1, lines 31-35, "Time is also saved as only the classes that are needed are loaded..." Method tables are well known in the art.

Per claim 10:

-memory is a heap memory of said computing system.

(Berry: Berry's invention disclosed loading code into a first (master) virtual machine. As such the memory is a heap memory. See FIG. 4, #245, Private Heap.)

Per claims 11 and 12:

-determining whether an internal representation of the class file exists in the virtual machine;

-creating an internal representation of the class file in the virtual machine when said determining determines that an internal representation of the class file does not exist in the virtual machine.

(Berry: Col. 3, lines 66-col. 4, line 12, "...a shared class is loaded into the second virtual machine by walking the class loader hierarchy...to determine for each class loader in the hierarchy whether it has previously loaded the class (determining whether an internal representation of the class file exists). This determination is performed...on the basis of said class loader cache...If the class has not been previously loaded...it is then determined whether the class has been loaded ...the client causes the class to be loaded (create an internal representation)" It is determined whether an internal representation of the class file exists else it is created.)

Per claim 15:

Berry disclosed a relationship between multiple virtual machines. Classes are loaded into the heap memory of a master virtual machine. Col. 2, lines 58-67, "The invention provides a master (first) virtual machine and at least one client (second) virtual machine running in parallel on the same computer system...certain class properties...may need to be set individually on each virtual machine." Thus Berry provides motivation for a "special version" of the class to be

loaded onto the client / second virtual machine. Berry disclosed (col. 11, lines 26-33), "...the requirement to mirror all of the method block is platform dependent..." Thus Berry suggested that the second virtual machine may load only a portion of the class. Berry disclosed:

-A method of loading a class file into a virtual machine, said class file being associated with a class, and said virtual machine operating in a computing system, said method comprising:
(Berry: col. 2, line 40, "...provides a method..." Also see FIG. 4. Abstract, lines 2-3, "A class file is loaded by the first virtual machine into shared memory...")

-determining whether said class file exists in a dedicated heap memory portion;
(Berry: Col. 6, lines 22-27, "For each class included within or referenced by a program, the JVM effectively walks up the class loaded hierarchy...to see if any class loader has previously loaded the class(determines whether said class file exists).")

-loading said class file in its entirety into said dedicated heap memory portion when said determining determines that said class file does not exist in said dedicated heap memory portion;
(Berry: Col. 6, lines 27-35, "If the response from all three class loaders is negative, the JVM walks back down the hierarchy, with the Primordial class loader first attempting to locate the class...the Extension class loader then make a similar attempt...Application class loader then tries to load the class...")

-determining whether an internal representation of said selected information exists in said virtual machine;

(Berry: Col. 3, lines 66-col. 4, line 8, "...a shared class is loaded into the second virtual machine by walking the class loader hierarchy...to determine for each class loader in the hierarchy whether it has previously loaded the class. This determination is performed...on the basis of said class loader cache...If the class has not been previously loaded...it is then determined whether the class has been loaded ..." It is determined whether an internal representation of the class file exists else it is created.)

-creating an internal representation of said selected information in said virtual machine when said determining determines that an internal representation of said selected information of said class file does not exist in said virtual machine;

(Berry: Col. 6, lines 22-33, "For each class included within or referenced by a program, the JVM effectively walks up the class loader...attempting to locate the class...tries to load the class..." The virtual machine will attempt to load (create an internal representation) if it does not exist.

Merrick provided additional information regarding the selection of information that is to be loaded into the virtual machine. Merrick disclosed:

-encountering a request to use at least one method of a class associated with a class file;

(Merrick: Col. 3, lines 5459, "...group the methods together if they were dependent on one another. This could be indicated in the class metadata where the 'compulsory methods' for downloading were referenced. The classloader would check for 'compulsory methods' in the metadata..." Also, col. 5, lines 16-17, "...one of the methods within the class was referenced then only the block of data representing this method is loaded...along with the other essential components of the class." Dependent methods (methods that request another method) can be grouped together.)

-loading into said virtual machine said selected information associated with said at least one method of said class and not loading into said virtual machine information that was not selected. (Merrick: Col. 4, lines 33-37, "...the modified class loader will be asked to load x.class (step 1) (loading said selected information). Instead of downloading the x.class in its entirety (not loading information which has not been selected)..." Merrick suggests that only the requested information is loaded. Col. 4, lines 22-27, "...a load method invoker (modified class loader) to retrieve the method component that has not been loaded onto the client..." and col. 4, lines 64-65, "The method byte code is written to the location pointed at by the invoker...")

-selecting information associated with said at least one method of said class, wherein said selecting operates to select said information form said class file in said dedicated memory;

(Merrick disclosed that the class file could be partitioned to allow for dependent methods to be grouped together for loading. Col. 3, lines 51-54, "Other ways of breaking down a class file are

possible, for instance one could group some methods together if they were dependent on one another.")

Therefore, it would have been obvious, to one of ordinary skill in the art, to have modified Berry's invention, to include the selection of information for a selective load of class files into a virtual machine, as further detailed by Merrick, because both references pertain to virtual machine loading of classes, and both disclose a modified load of class code for the purpose of a quicker virtual machine start up. Berry, col. 11, lines 40-41. Merrick, col. 1, lines 31-35, "Time is also saved as only the classes that are needed are loaded..."

Per claim 18:

-wherein said internal representation includes a reference cell associated with said at least one method.

(Berry: Col. 6, line 65- col. 7, line 12, "The class storage area further includes a method block area, which is used to store information relating to the code, such as invokers, and a pointer to the code, which may for example be in method code area...Classes stored as objects in the heap contain a reference to their associated data such as method byte code etc. in class storage area..." The internal representation includes references to associated methods.)

8. Claims 13, 14, and 16 are rejected under 35 U.S.C. 103(a) as being unpatentable over US Patent 6,738,977 to Berry et al., in view of US Patent 6,339,841 to Merrick et al., and further in view of "EJVM: an economic JAVA run-time environment of embedded devices", by Da-Wei Chang and Ruei-Chuan Chang.

Berry disclosed a class loader that loaded from a heap memory area. Berry suggested that the class could be modified for the subsequent loading. Merrick disclosed a modified class loader that loaded only selected information into a virtual machine. While Berry disclosed garbage collection, the combination failed to disclose information regarding removing class files from a memory portion using a Least Recently Used policy.

However, Chang and Chang disclosed "removing said class file / on a Least Recently Used basis /from said memory portion" on pages 140-143. Garbage collection is a well known feature of virtual machines. Using the Least Recently Used policy for determining which code segments to remove is a reasonable technique. It is well known in the art as criteria for choosing selected information. Chang and Chang split class files and installed them into a cache. They benchmarked the LRU policy for various sized caches. See fig. 11, page 142.

Therefore, it would have been obvious, to one of ordinary skill in the art, at the time of the invention, to have modified Berry's and Merrick's combined invention to also address garbage collection policies, including LRU, because virtual machine efficiency can be enhanced by minimizing memory requirements, and selecting and clearing unused code is a well known technique.

Response to Arguments

9. Applicant has argued, in substance, the following:

(A) As Applicant has pointed out on page 6, paragraph (a), of Remarks dated 09/20/2004, "Merrick does not teach or suggest selecting information from a class file in a memory portion after the class file has been loaded in its entirety into the memory portion."

Examiner's Response: Examiner disagrees. The Berry reference suggests first loading class files in their entirety into a master JVM memory portion / heap. To be followed by loading a selected portion of the master memory portion into a client JVM memory portion / heap. See response to claim 1 above.

(B) As Applicant has pointed out on page 6, paragraph (b), "Merrick cannot be properly combined with Berry et al."

Examiner's Response: Examiner disagrees. Both prior art references deal with improving efficiency of a virtual machine (Berry: col. 2, lines 61) (Merrick: col. 1, lines 31-33) by modifying classes that are loaded into a virtual machine memory portion / heap. Both references suggest loading a subset of classes. Therefore, it is obvious, to combine these references.

(C) As Applicant has pointed out on page 7, paragraph (c), "The combination of the Merrick and Berry do not teach or suggest several other features..." (the features of claims 5, 7, and 9?)

Examiner's Response: Examiner disagrees. See rejection of claims 5, 7, and 9 above.

(D As Applicant has pointed out on page 7, paragraph (c), "...the method table of Merrick does not teach or suggest a method reference portion that includes one or more references cells which provide information associated with one or more methods which have been selected to be loaded into the virtual machine."

Examiner's Response: Examiner disagrees. Merrick discloses a method table. Method tables are well known in the art. See supplemental reference from JavaWorld, "How the JAVA virtual machine handles method invocation and return." Method tables at least provide the operand, operators, and description (provide information associated with method) in reference cells.

Examiner maintains the rejections of claims 1, 4-19 and 21.

Conclusion

10. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

Also note:

US Patent 6,202,208 to Holiday, Jr. – Modifying a loader environment of the JVM by altering the existing method body with a patch then loading. See Abstract, line 6, col. 7, line 51-col. 8, line 21 and figs. 3 & 4.

US Patent 6,072,953 to Cohen et al.- Extended class loader may apply any modification to classes.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Mary Steelman, whose telephone number is (571) 272-3704. The examiner can normally be reached Monday through Thursday, from 7:00 AM to 5:30 PM If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Q. Dam can be reached at (571) 272-3695. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Mary Steelman

01/19/2005

WEI Y. ZHEN
PRIMARY EXAMINER